1. Comparison of CNN and Yolo

CNN (Convolutional Neural Network) and YOLO (You Only Look Once) are two popular deep learning-based algorithms used for computer vision tasks, particularly in object detection. While both methods serve the same purpose, they differ in terms of architecture, performance, and application. Let's compare CNN and YOLO in detail:

1. Architecture:
-CNN is an example of a feed-forward neural network with a multi-layered architecture that includes convolutional, pooling, and fully connected layers. By applying filters to the input image, the convolutional layers conduct feature extraction, capturing spatial hierarchies of features.
- YOLO: This single-stage object detection technique divides the input image into a grid and forecasts the bounding boxes and class probabilities in advance. To produce object detections, it employs a succession of convolutional layers followed by fully linked layers.

2. Object Detection:
- CNN: Traditional CNN architectures, such as VGGNet, ResNet, or Inception, are not specifically designed for object detection. Instead, they are primarily used for image classification and require additional modifications, like sliding window or region proposal methods, to perform object detection.
- YOLO: YOLO is specifically designed for object detection tasks. It directly predicts bounding box coordinates and class probabilities for multiple objects in a single pass through the network. YOLO divides the image into a grid and assigns each grid cell responsibility for detecting objects within it.

3. Speed and Efficiency:
- CNN: CNN-based object recognition approaches that employ sliding windows or region proposal techniques can be computationally expensive because they call for iteratively applying the model to various parts of the image. This method can be cumbersome, particularly when working with huge photos or a lot of items.
- YOLO: YOLO is renowned for its performance in real-time. YOLO delivers high efficiency by avoiding the requirement for numerous passes on various image regions. It can process images in real time or very close to real time because it does detection in a single pass. YOLO is frequently used in applications like video surveillance that need for quick and precise object recognition.

4. Accuracy and Localization:
- CNN: CNN-based object detectors, when combined with region proposal or sliding window methods, can achieve high accuracy in object detection. However, they may struggle with accurately localizing small or densely packed objects due to the limitations of region proposal methods.

- YOLO: YOLO sacrifices some accuracy for speed and efficiency. While YOLO performs well for objects with regular shapes and larger sizes, it may struggle with precise localization of small objects or objects with complex structures. However, newer versions of YOLO, such as YOLOv4 and YOLOv5, have made significant improvements in accuracy while maintaining real-time performance.

5. Application and Use Cases:
- CNN: CNNs are widely used for various computer vision tasks, including image classification, object recognition, and semantic segmentation. They are suitable for scenarios where high accuracy is the primary requirement, even if the inference speed is not critical.
- YOLO: YOLO's real-time performance makes it well-suited for applications where fast object detection is crucial, such as autonomous driving, robotics, and real-time surveillance. YOLO's ability to process video streams efficiently enables applications that require tracking and detection in real-time environments.

2. Different layers in CNN and your understanding

There are numerous levels in a standard CNN (Convolutional Neural Network) architecture, and each layer applies a different operation to the incoming data. Let's examine the many levels typically present in a CNN:

1. Input Layer: The input layer shows the unprocessed image data or a preceding layer's output. It keeps track of the input image's pixel values and sends them further through the network.

2. Convolutional Layer: The convolutional layer transforms the input image using a number of learnable filters (also known as kernels). Convolution is performed by each filter, which computes dot products with the local receptive field at each position as it moves spatially over the input. This method aids in the extraction of various visual elements like edges, corners, or textures.

3. Activation Layer: The network is given non-linearities by the activation layer. The output of the convolutional layer receives an element-by-element application of an activation function. ReLU (Rectified Linear Unit), the sigmoid function, and tanh are common activation functions. Due to its computational effectiveness and capacity to solve the vanishing gradient problem, ReLU is frequently utilised in CNNs.

4. Pooling Layer: The spatial dimensions of the feature maps produced by the convolutional layer are downsampled by the pooling layer. It lowers the feature maps' resolution while retaining crucial details. Max pooling (choosing the highest value in each pooling region) and average pooling (taking the average value in each region) are two common pooling techniques.

5. Batch Normalisation Layer: This layer applies a mini-batch of samples to normalise the activations of the preceding layer. The training process is stabilised, convergence is accelerated, and internal covariate shift is reduced.

6. Fully Connected Layer: This layer connects all of the neurons in the current layer to all of the neurons in the layer below it. It carries out a linear transformation after which an activation function is applied. The fully connected layers at the end of the CNN architecture are in charge of translating high-level features into probabilities for classes or regression coefficients.

7. Flattening Layer: This layer creates a one-dimensional vector from the multidimensional output of the previous layer. The data can be supplied into a layer that is completely connected thanks to this flattening procedure.

8. Dropout Layer: During training, the dropout layer randomly changes a portion of the input neurons to 0. By decreasing the dependency of neurons, it helps minimise overfitting and enables the network to acquire more robust characteristics.